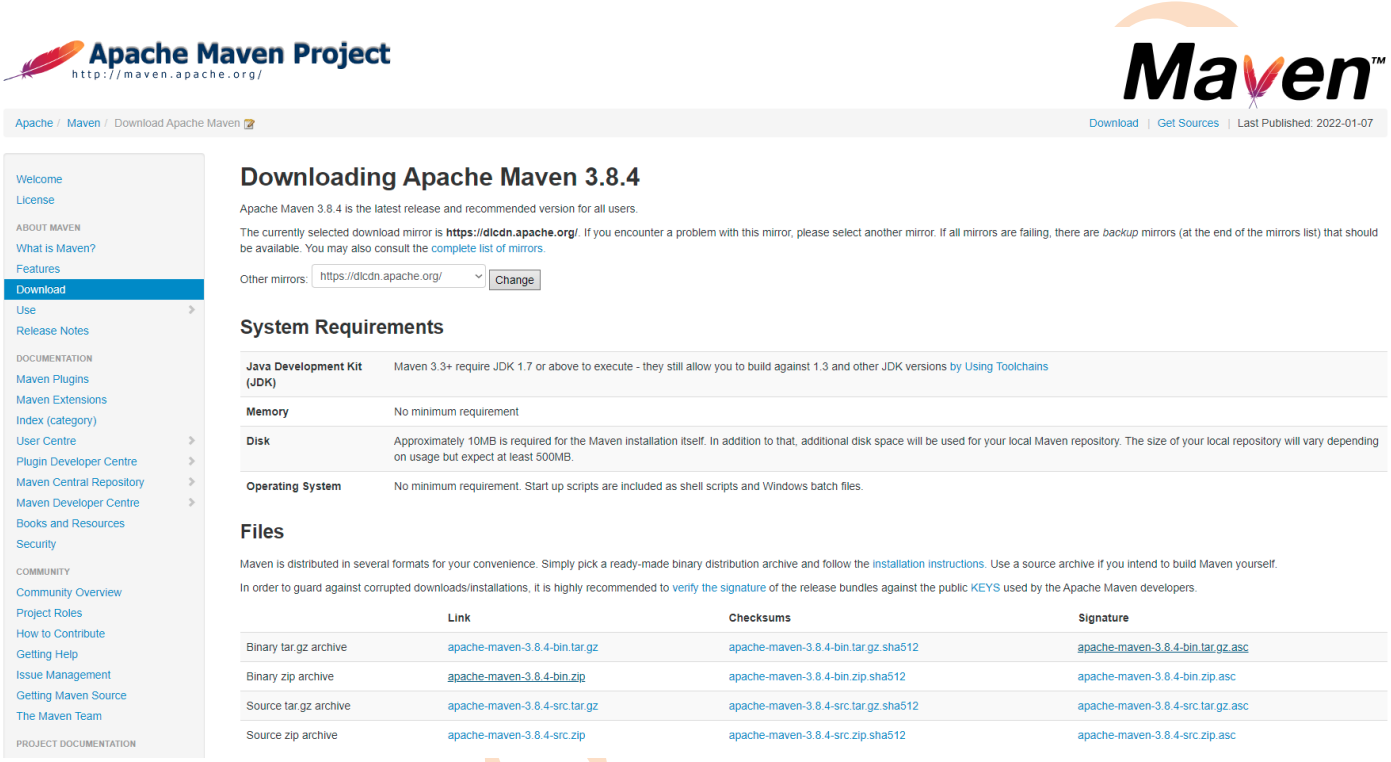
Задание №3 Сборка проекта на основе Maven

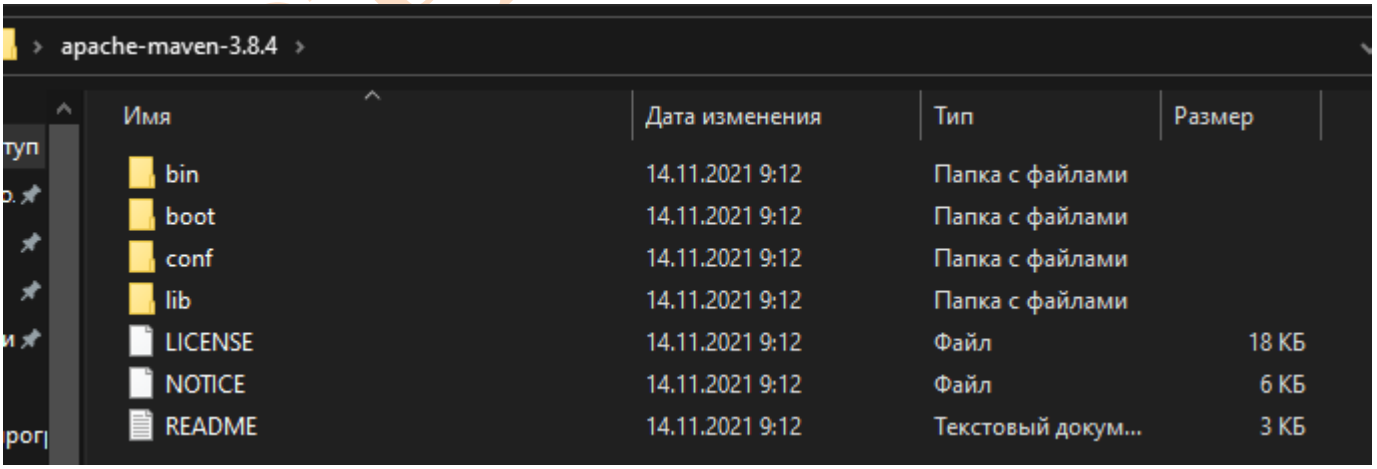
1) Ps создайте папочку maven, куда будем сгружать собранные примеры, что бы не потерялись. Так же можете сохранять их в этот файл в виде текста из конфигурационных файлов.

2) *Установите maven. Настройте environment variables и проверьте, что maven настроен корректно.*

Скачайте maven архив binary с http://maven.apache.org/download.cgi



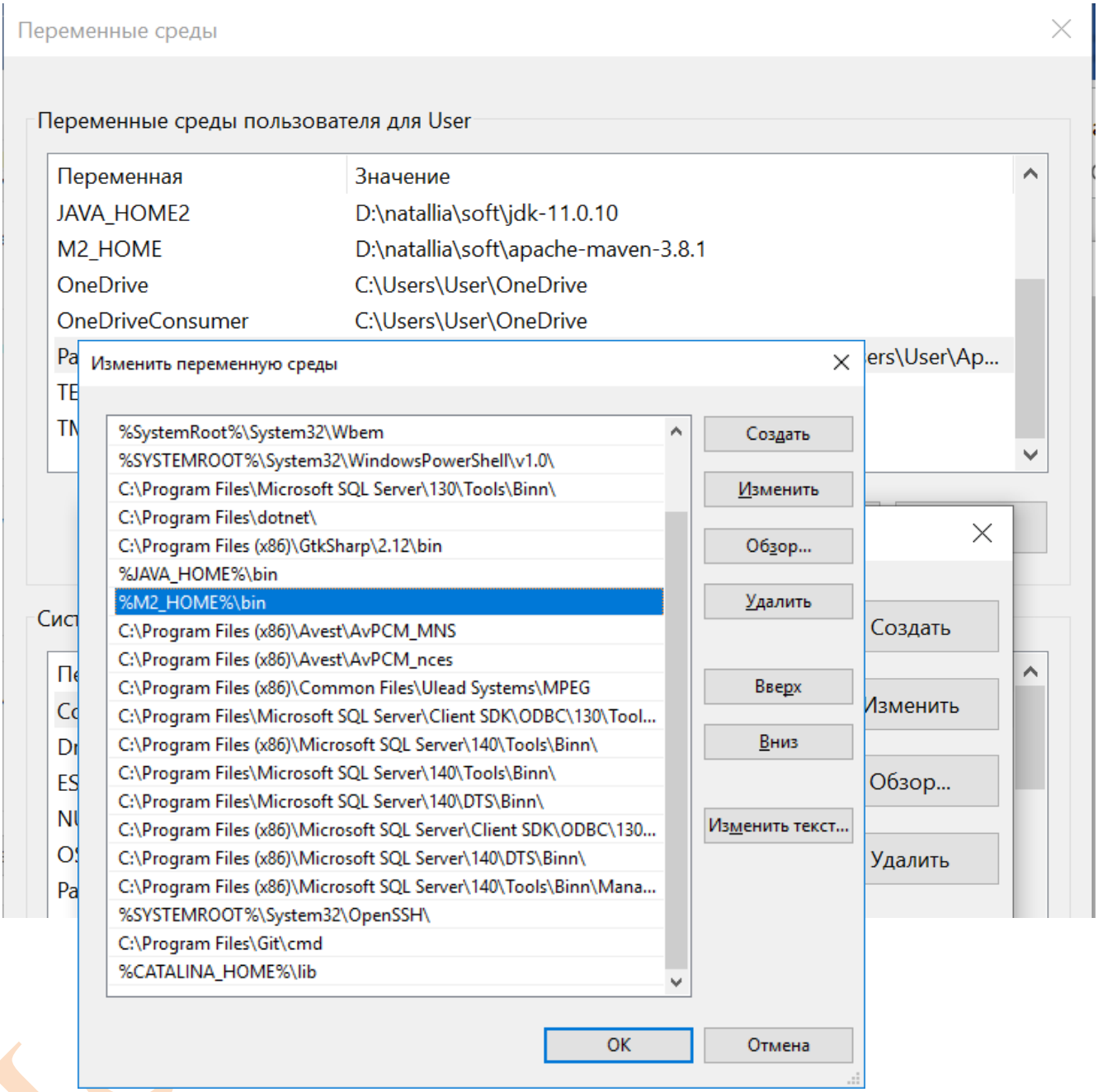
Обратите внимание что подбиться версия JDK 1.7 и выше. После загрузки вы получите zip архив около 10 Мб.



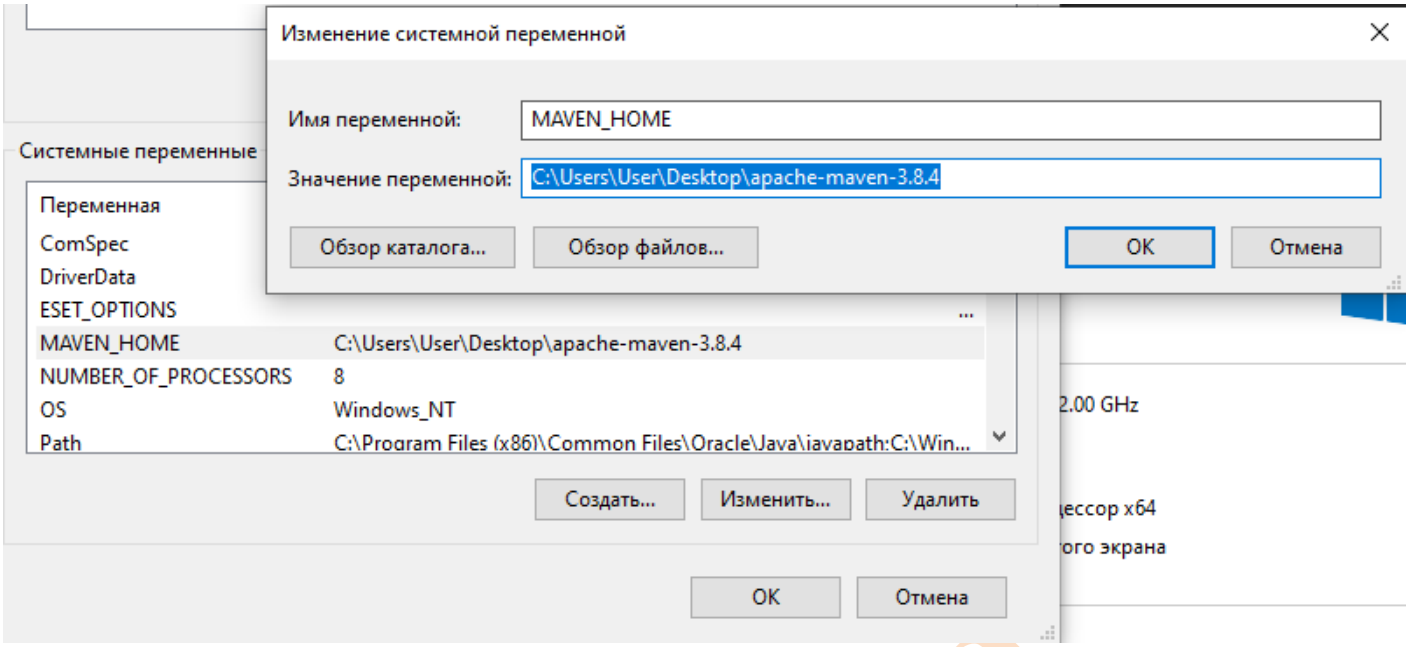
Maven устанавливается просто копированием в нужную директорию — никакого инсталлера нет. Распакуйте архив в любую папку. Для Windows, как правило, путь к папке не должен содержать пробелов.

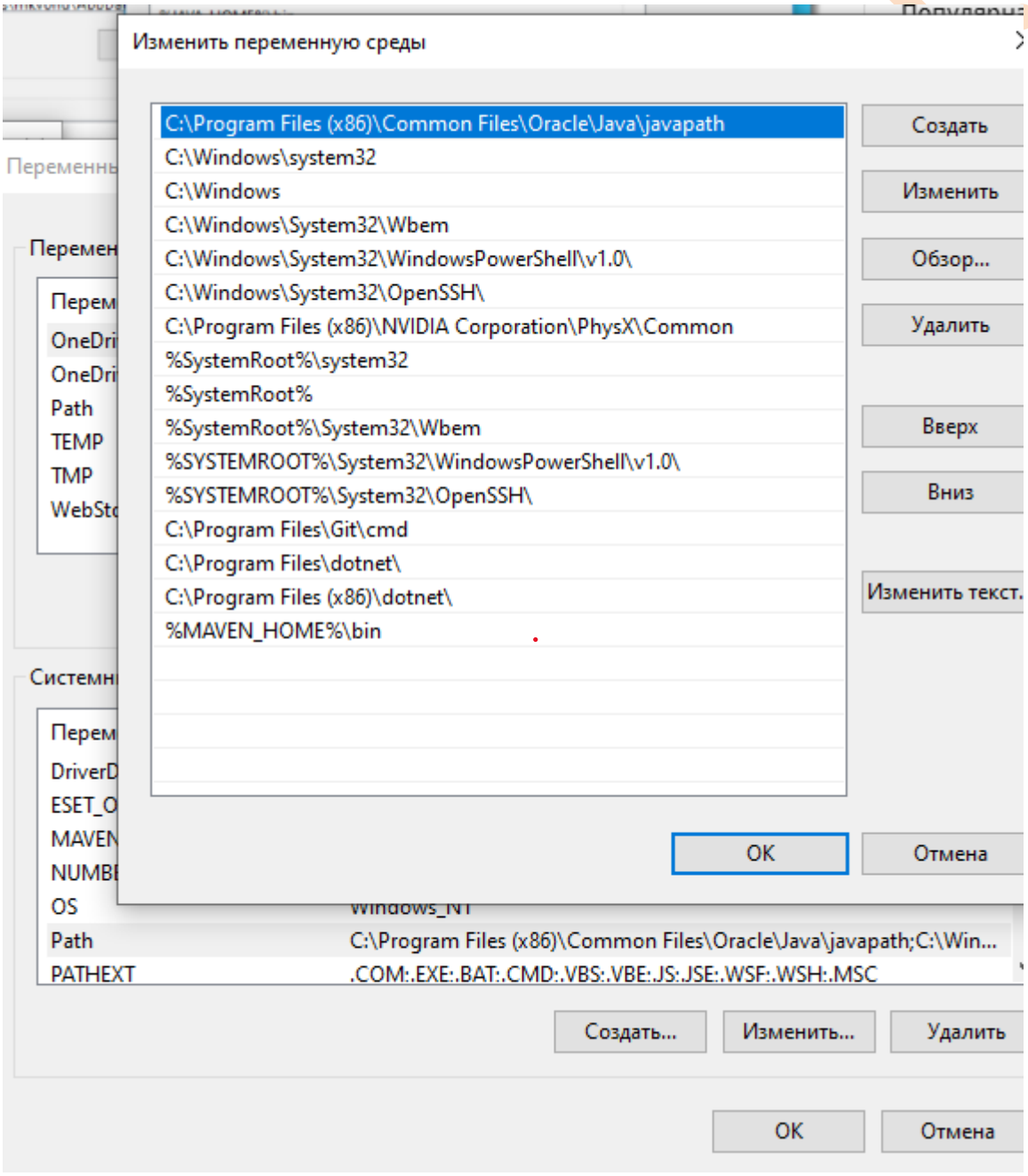
Для начала использования Maven необходимо настроить переменные среды Windows. MAVEN_HOME (M2_HOME) переменная должна быть установлена и

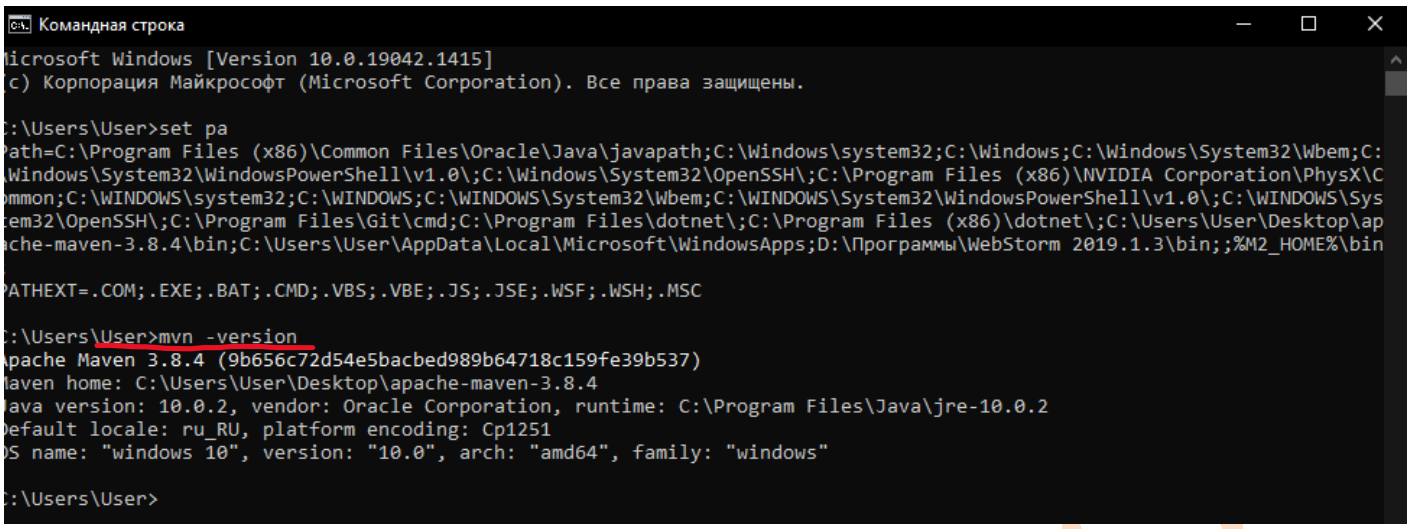
PATH переменная должна быть модифицирована для включения папки, откуда запускается Maven.



Убедитесь, что они установлены:


Изменение системной переменной
Имя переменной:
MAVEN_HOME
Значение переменной:
C:\Users\User\Desktop\apache-maven-3.8.4
Обзор каталога...
Обзор файлов...
OK
Отмена
Системные переменные
Переменная
ComSpec
DriverData
ESET_OPTIONS
MAVEN_HOME
C:\Users\User\Desktop\apache-maven-3.8.4
NUMBER_OF_PROCESSORS
8
OS
Windows_NT
Path
Создать...
Изменить...
Удалить
OK
Отмена

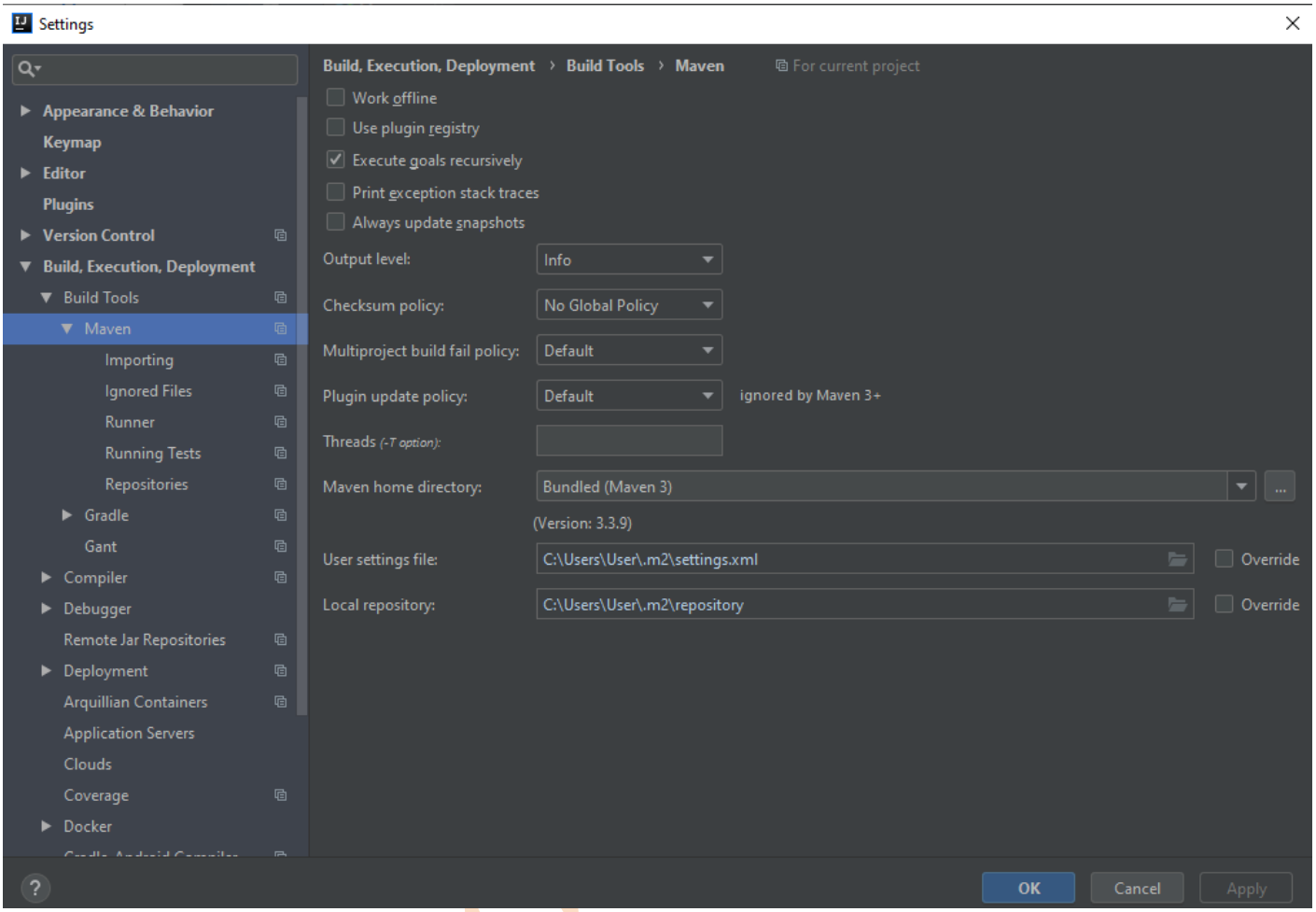

Изменить переменную среды
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
C:\Windows\system32
C:\Windows
C:\Windows\System32\Wbem
C:\Windows\System32\WindowsPowerShell\v1.0\
C:\Windows\System32\OpenSSH\
C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files\dotnet\
C:\Program Files (x86)\dotnet\
%MAVEN_HOME%\bin
Создать
Изменить
Обзор...
Удалить
Вверх
Вниз
Изменить текст...
OK
Отмена
Path
C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\Win...
PATHEXT
.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
Создать...
Изменить...
Удалить
OK
Отмена



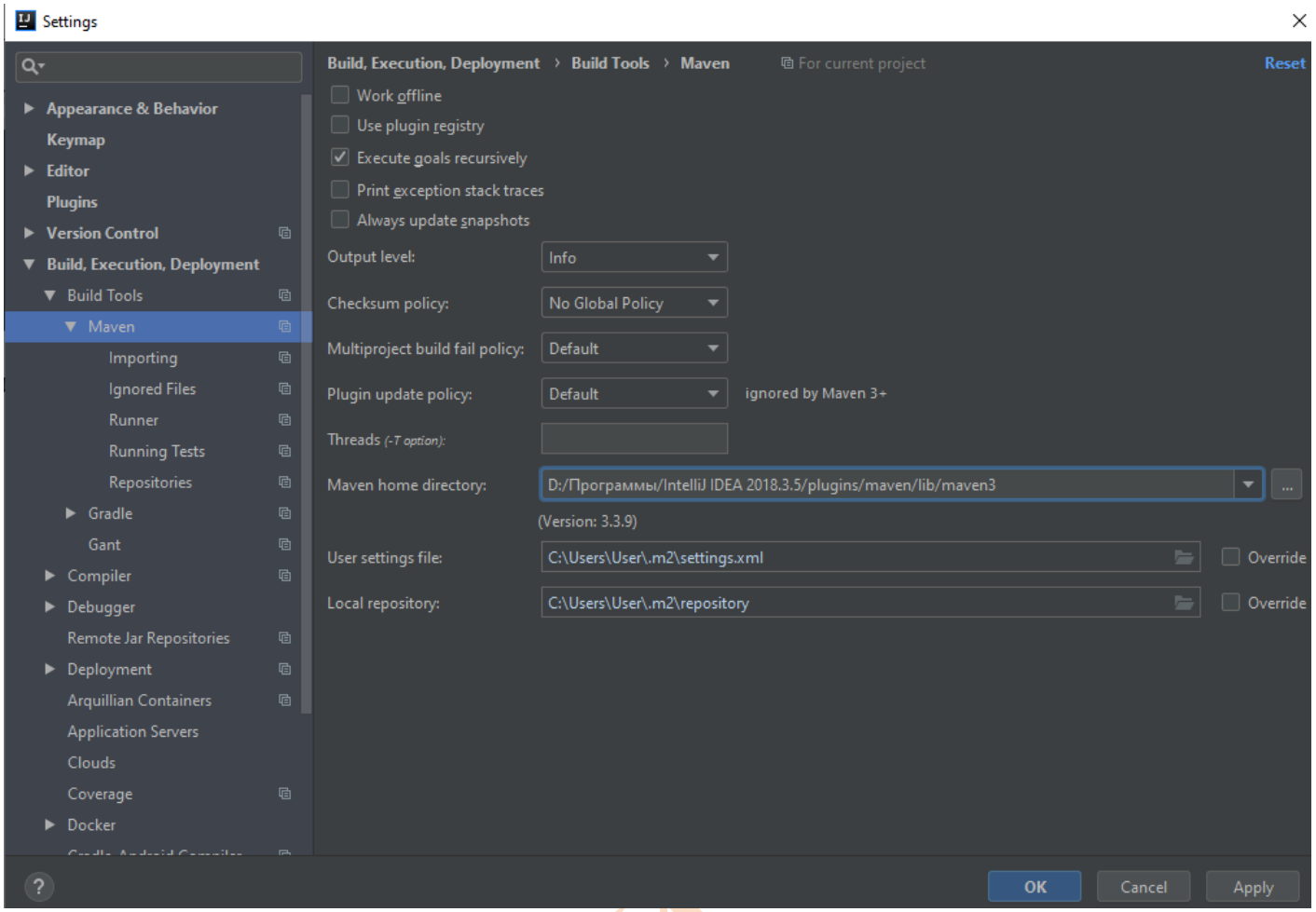
ApacheMaven, теперь, готов к использованию. Он, также, доступен для интеграции с IDE и другими программными средствами, предназначенными для разработки.mvn --version

3) Проверка поддержки Maven

Шаблон, на основе которого был создан проект включает поддержку maven

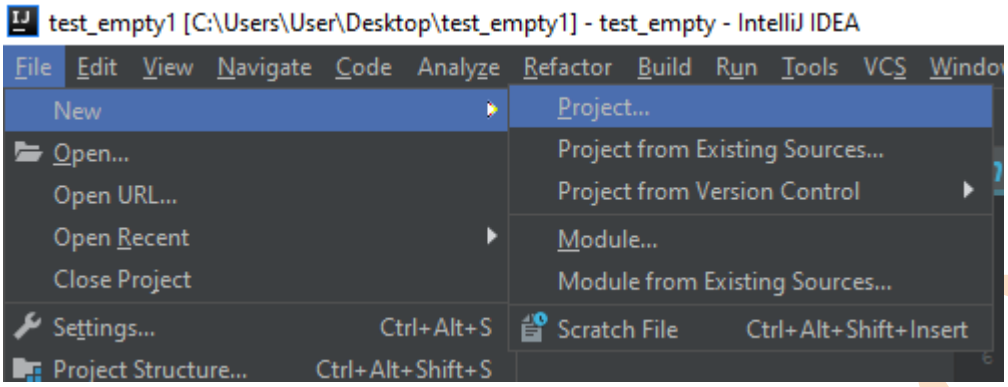


Если нужно будет поменять директорий maven или репозиторий это также можно сделать здесь.

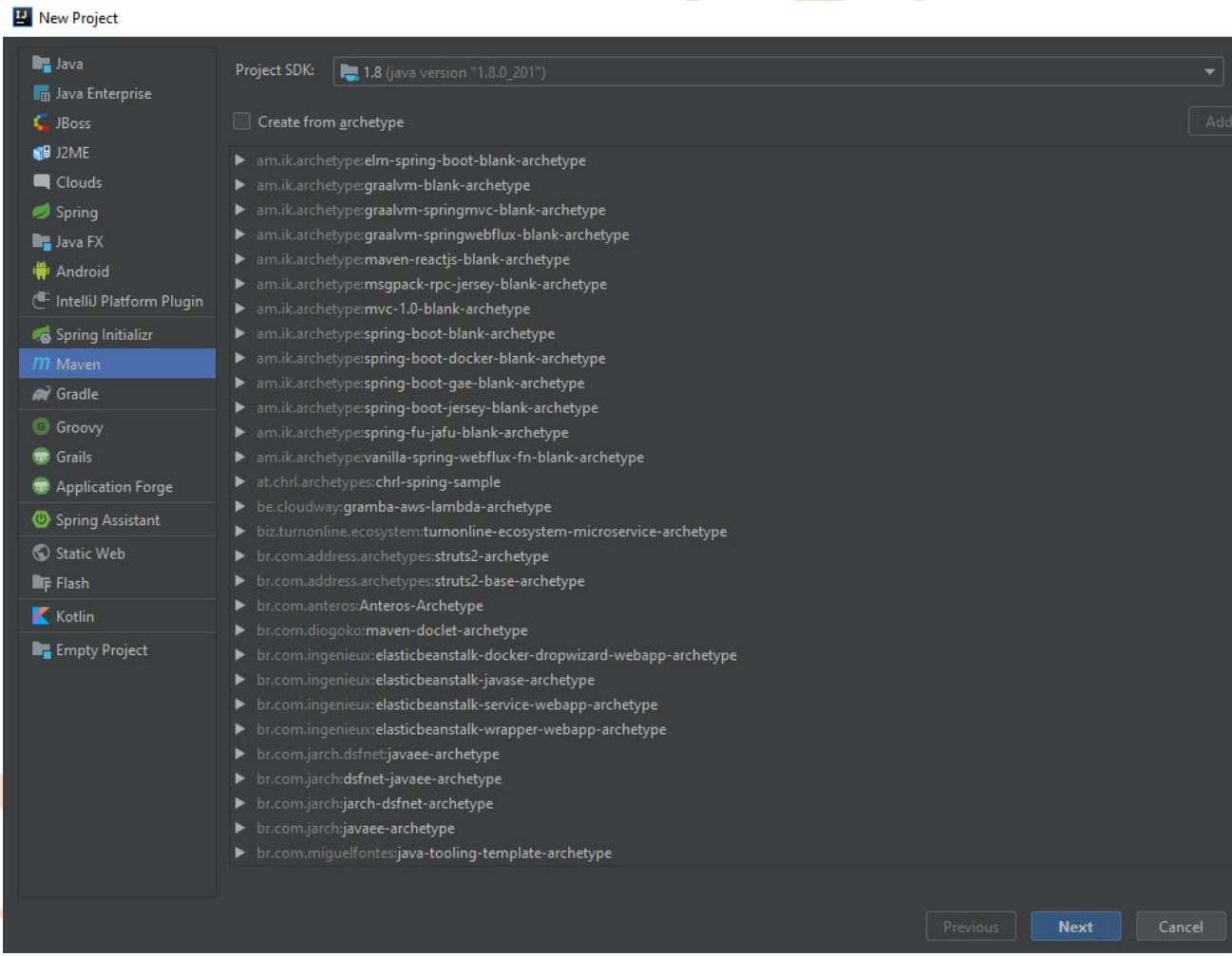

Settings
Build, Execution, Deployment › Build Tools › Maven
For current project
Reset
Work offline
Use plugin registry
Execute goals recursively
Print exception stack traces
Always update snapshots
Output level:
Info
Checksum policy:
No Global Policy
Multiproject build fail policy:
Default
Plugin update policy:
Default
ignored by Maven 3+
Threads (-T option):
Maven home directory:
D:/Программы/IntelliJ IDEA 2018.3.5/plugins/maven/lib/maven3
(Version: 3.3.9)
User settings file:
C:\Users\User\.m2\settings.xml
Override
Local repository:
C:\Users\User\.m2\repository
Override
Appearance & Behavior
Keymap
Editor
Plugins
Version Control
Build, Execution, Deployment
Build Tools
Maven
Importing
Ignored Files
Runner
Running Tests
Repositories
Gradle
Gant
Compiler
Debugger
Remote Jar Repositories
Deployment
Arquillian Containers
Application Servers
Clouds
Coverage
Docker
OK
Cancel
Apply

4) *Интеграция Maven*

Создайте проект с помощью Maven.



После чего



Выбираем Maven

Архетипы и прочее пока что не трогаем. Жмем далее.

Задаем имя проекта. А также спецификацию.

После чего должен быть автоматически создан проект с pom.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>test_empty</groupId>
    <artifactId>test_empty</artifactId>
    <version>1.0-SNAPSHOT</version>
    <name>Servlet</name>
    <packaging>war</packaging>

    <properties>
        <maven.compiler.target>1.8</maven.compiler.target>
        <maven.compiler.source>1.8</maven.compiler.source>
        <junit.version>5.7.0</junit.version>
    </properties>

</project>
```

Изучите раздел с существующими зависимостями. У вас должно быть примерно так после добавления:

.

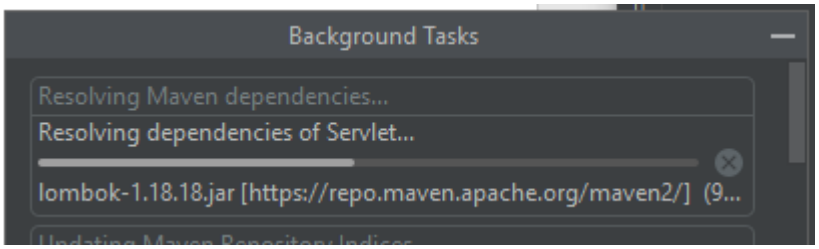
```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
                <execution>
                    <id>id.pre-clean</id>
                    <phase>pre-clean</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>pre-clean phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    <id>id.clean</id>
                    <phase>clean</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>clean phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    <id>id.post-clean</id>
                    <phase>post-clean</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>post-clean phase</echo>
```

```
                        </tasks>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Далее вы сможете увидеть добавление зависимостей в проект.



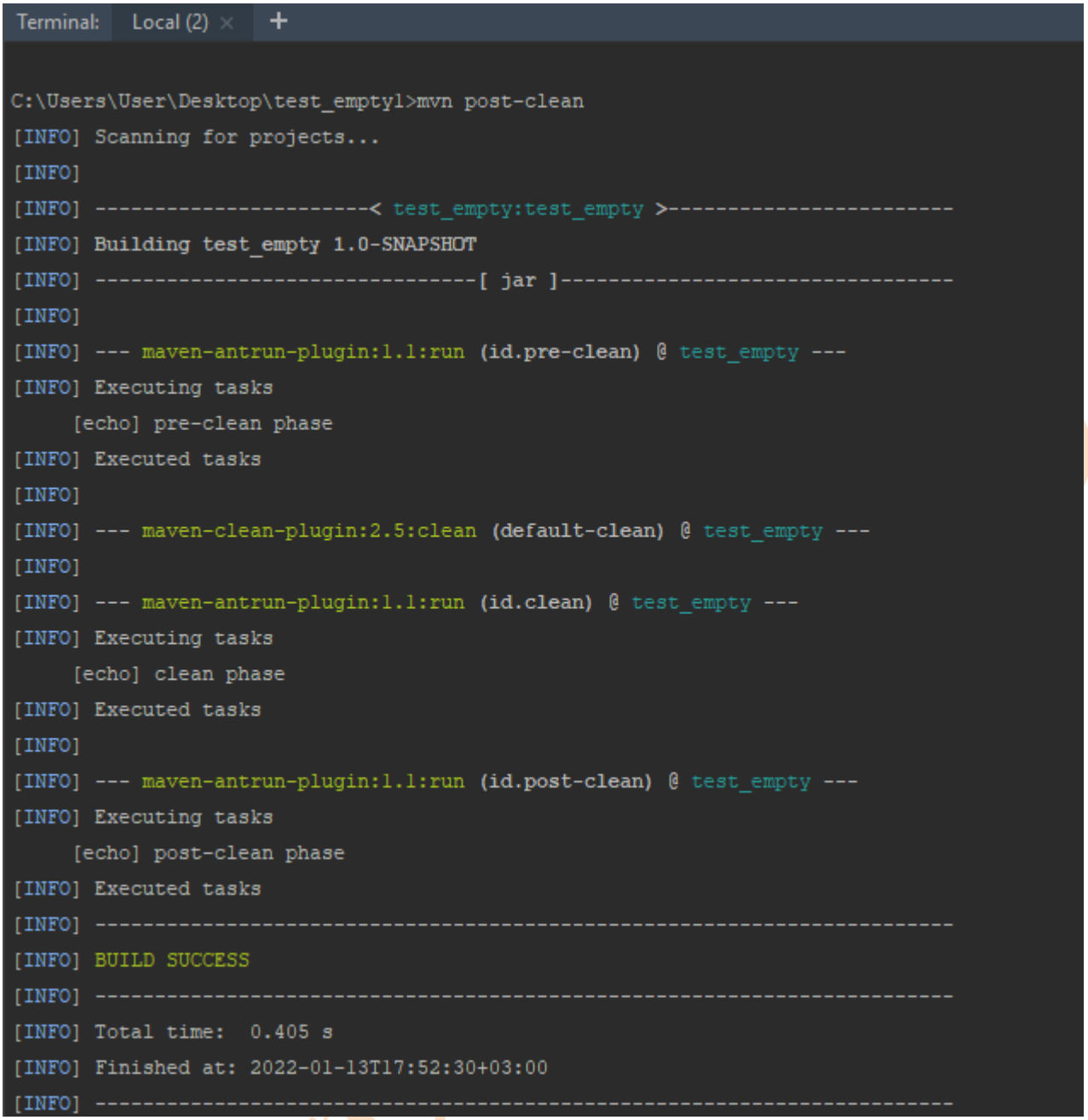
Проверьте в консоли команду:

```
Terminal:  Local (2)  +
Microsoft Windows [Version 10.0.19042.1415]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\User\Desktop\test_empty1>mvn -version
Apache Maven 3.8.4 (9b656c72d54e5bacbed989b64718c159fe39b537)
Maven home: C:\Users\User\Desktop\apache-maven-3.8.4
Java version: 10.0.2, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jre-10.0.2
Default locale: ru_RU, platform encoding: Cp1251
OS name: "windows 10", version: "10.0", arch: "amd64", family: "windows"

C:\Users\User\Desktop\test_empty1>
```

После чего выполните команду:



Maven начнет обработку и отображение всех фаз чистого жизненного цикла.

Жизненный цикл по умолчанию (или сборка)

Это основной жизненный цикл Maven, который используется для создания приложения. Он имеет следующие 23 фазы.

Sr.No.	Фаза жизненного цикла и описание
validate	Подтверждает, является ли проект корректным и вся ли необходимая информация доступа для завершения процесса сборки.

initialize	Инициализирует состояние сборки, например, различные настройки.
generate-sources	Включает любой исходный код в фазу компиляции.
process-sources	Обрабатывает исходный код (подготавливает). Например, фильтрует определённые значения.
generate-resources	Генерирует ресурсы, которые должны быть включены в пакет.
process-resources	Копирует и отправляет ресурсы в указанную директорию. Это фаза перед упаковкой.
compile	Комплирует исходный код проекта.
process-classes	Обработка файлов, полученных в результате компиляции. Например, оптимизация байт-кода Java классов.
generate-test-sources	Генерирует любые тестовые ресурсы, которые должны быть включены в фазу компиляции.
process-test-sources	Обрабатывает исходный код тестов. Например, фильтрует значения.
test-compile	Компилирует исходный код тестов в указанную директорию тестов.
process-test-classes	Обрабатывает файлы, полученные в результате компиляции исходного кода тестов.
test	Запускает тесты, используя приемлемый фреймворк юнит-тестирования (например, Junit).
prepare-package	Выполняет все необходимые операции для подготовки пакет, непосредственно перед упаковкой.
package	Преобразует скомпилированный код и пакет в дистрибутивный формат. Такие как JAR, WAR или EAR.
pre-integration-test	Выполняет необходимые действия перед выполнением интеграционных тестов.
integration-test	Обрабатывает и распаковывает пакет, если необходимо, в среду, где будут выполняться интеграционные тесты.

post-integration-test	Выполняет действия, необходимые после выполнения интеграционных тестов. Например, освобождение ресурсов.
verify	Выполняет любые проверки для подтверждения того, что пакет пригоден и отвечает критериям качества.
install	Устанавливает пакет в локальный репозиторий, который может быть использован как зависимость в других локальных проектах.
deploy	Копирует финальный пакет (архив) в удалённый репозиторий для, того, чтобы сделать его доступным другим разработчикам и проектам.

Теперь обновим pom файл и разберем, в чем он нам поможет сейчас:

Необходимо уточнить два момента:

Когда мы выполняем команду Maven, например install, то будут выполнены фазы до install и фаза install.

Различные задачи Maven будут привязаны к различным фазам жизненного цикла Maven в зависимости от типа архива (JAR/WAR/EAR).

В следующем примере, мы привязываем задачу maven-antrun-plugin:run к нескольким фазам жизненного цикла сборки. Это также позволяет нам вызывать текстовые сообщения, отображая фазу жизненного цикла.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
                <execution>
                    <id>id.validate</id>
                    <phase>validate</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>validate phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    <id>id.compile</id>
                    <phase>compile</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>compile phase</echo>
                        </tasks>
                    </configuration>
                </execution>
```

```
                <execution>
                    <id>id.test</id>
                    <phase>test</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>test phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    <id>id.package</id>
                    <phase>package</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>package phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    <id>id.deploy</id>
                    <phase>deploy</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>deploy phase</echo>
                        </tasks>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

```
C:\Users\User\Desktop\test_empty1>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< test_empty:test_empty >-----------------------
[INFO] Building test_empty 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.validate) @ test_empty ---
[INFO] Executing tasks
     [echo] validate phase
[INFO] Executed tasks
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ test_empty ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 0 resource
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ test_empty ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.compile) @ test_empty ---
[INFO] Executing tasks
     [echo] compile phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.733 s
[INFO] Finished at: 2022-01-13T18:47:05+03:00
[INFO] ------------------------------------------------------------------------

C:\Users\User\Desktop\test_empty1>
```

Жизненный цикл Site

Плагин Maven Site обычно используется для создания свежей документации, создания отчетов, развертывания сайта и т. Д. Он имеет следующие фазы:

Он включает в себя такие фазы:

- pre-site
- site
- post-site
- site-deploy

Изменим наш pom файл:

В примере ниже мы прикрепляем задачу maven-antrun-plugin:run ко всем фазам жизненного цикла Site. Это позволяет нам вызывать текстовые сообщения для отображения фаз жизненного цикла.

```
<build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-antrun-plugin</artifactId>
            <version>1.1</version>
            <executions>
                <execution>
                    <id>id.pre-site</id>
                    <phase>pre-site</phase>
                    <goals>
                        <goal>run</goal>
                    </goals>
                    <configuration>
                        <tasks>
                            <echo>pre-site phase</echo>
                        </tasks>
                    </configuration>
                </execution>

                <execution>
                    (Добавить id.site)

                </execution>

                <execution>
                    (Добавить id.post-site)

                </execution>

                <execution>
                    (Добавить id.site-deploy)

                </execution>

            </executions>
        </plugin>
    </plugins>
</build>
```

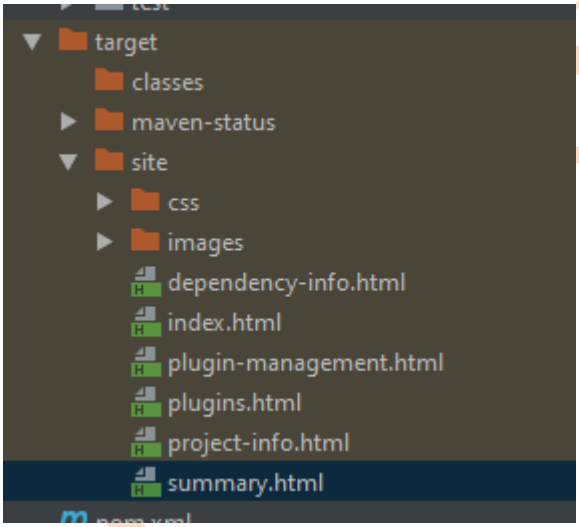
После чего выполним команду:

```
C:\Users\User\Desktop\test_empty1>mvn site
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< test_empty:test_empty >----------------------
[INFO] Building test_empty 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.pre-site) @ test_empty ---
[INFO] Executing tasks
     [echo] pre-site phase
[INFO] Executed tasks
[INFO]
```

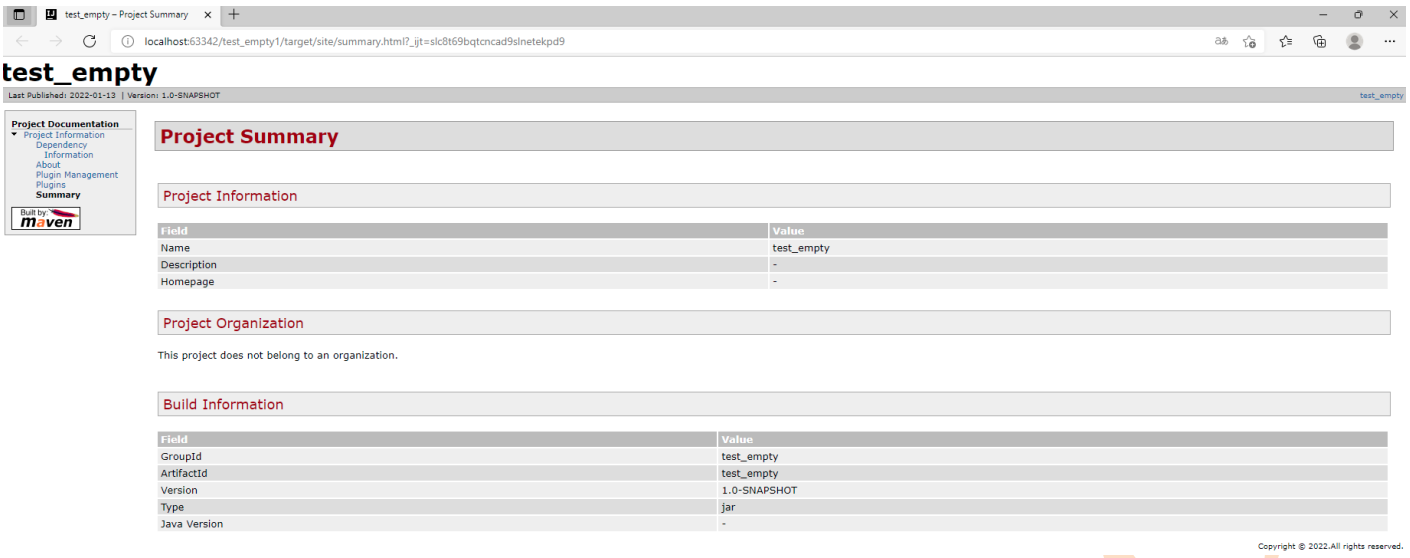
Если не работает, то исправим вот так:

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-site-plugin</artifactId>
    <version>3.7.1</version>
</plugin>
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-project-info-reports-plugin</artifactId>
    <version>3.0.0</version>
</plugin>
```

После чего в таргет выгрузилась информация по проекту:



Выбираем Нужную нам и жмякаем на просмотр:



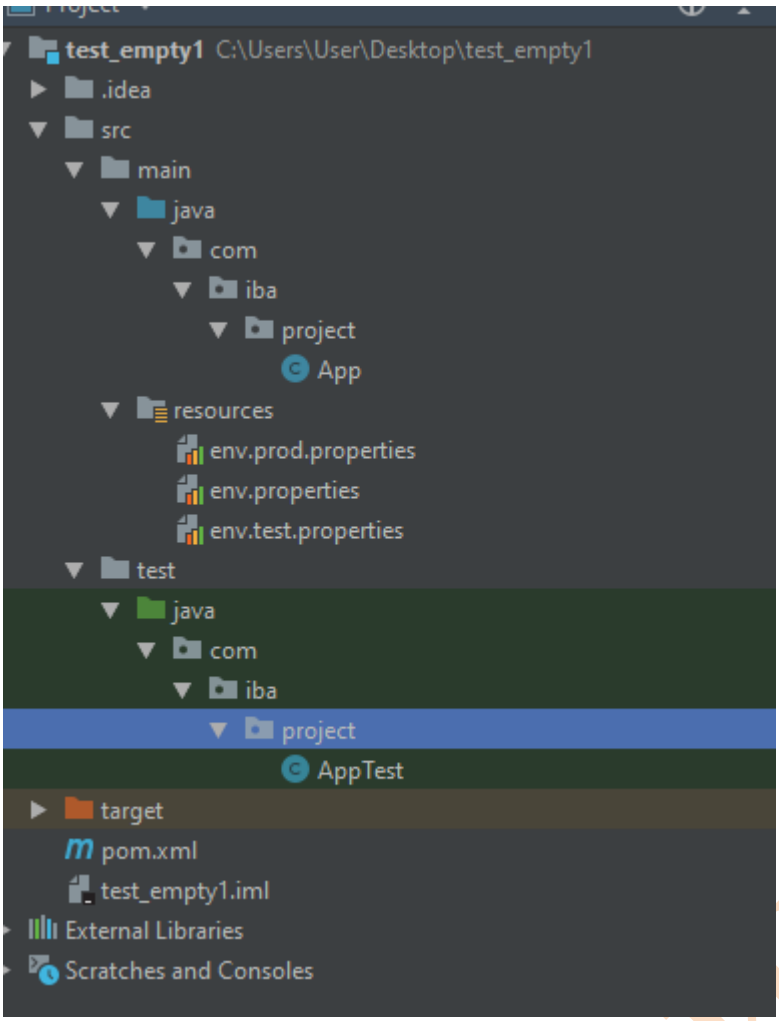
Теперь можно просто и легко делать верификацию наших проектов.
Перейдем к более сложным аспектам.

Maven — Создание профилей

Что такое профиль сборки?

Профиль сборки — это набор значений конфигурации, которые можно использовать для установки или переопределения значений по умолчанию сборки Maven. Используя профиль сборки, вы можете настроить сборку для различных сред.

Создадим структуру проекта:



Теперь в **каталоге src / main / resources** есть три специфичных для среды файла:

Sr.No.	Имя файла и описание
1	**env.properties** используется конфигурация по умолчанию, если профиль не указан.
2	**env.test.properties** тестовая конфигурация при использовании тестового профиля.
3	**env.prod.properties** производственная конфигурация при использовании профиля prod.

Явная активация профиля

В следующем примере мы добавим maven-antrun-plugin: запустите цель, чтобы проверить фазу. Это позволит нам отображать текстовые сообщения для разных профилей. Мы будем использовать pom.xml для определения различных профилей и активировать профиль в командной консоли с помощью команды maven.

```
<profiles>
    <profile>
        <id>test</id>
        <build>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-antrun-plugin</artifactId>
                    <version>1.1</version>
                    <executions>
                        <execution>
                            <phase>test</phase>
                            <goals>
                                <goal>run</goal>
                            </goals>
                            <configuration>
                                <tasks>
                                    <echo>Using env.test.properties</echo>
                                    <copy file="src/main/resources/env.test.properties"

tofile="${project.build.outputDirectory}/env.test.properties"/>
                                </tasks>
                            </configuration>
                        </execution>
                    </executions>
                </plugin>
            </plugins>
        </build>
    </profile>
```

В файл properties добавьте что-нибудь.
В результате наши в сбоку проекта будут включены данные настройки, указанные в тестовом файле или файле продакшина.

```
C:\Users\User\Desktop\test_empty1>mvn test -Ptest
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< test_empty:test_empty >-----------------------
[INFO] Building test_empty 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ test_empty ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ test_empty ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1251, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Users\User\Desktop\test_empty1\target\classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ test_empty ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\Users\User\Desktop\test_empty1\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ test_empty ---
[INFO] Changes detected - recompiling the module!
[WARNING] File encoding has not been set, using platform encoding Cp1251, i.e. build is platform dependent!
[INFO] Compiling 1 source file to C:\Users\User\Desktop\test_empty1\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ test_empty ---
[INFO] Surefire report directory: C:\Users\User\Desktop\test_empty1\target\surefire-reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running com.iba.project.AppTest
Tests run: 0, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.002 sec
```

Самостоятельно сделайте для продакшина и нормального билда проекта.
Для этого создайте ряд профилей: normal и prod.
Измените goal в каждом. Измените id. Измените echo.
Запуск каждого профиля начинается аналогично примеру выше, далее –P{id профиля}

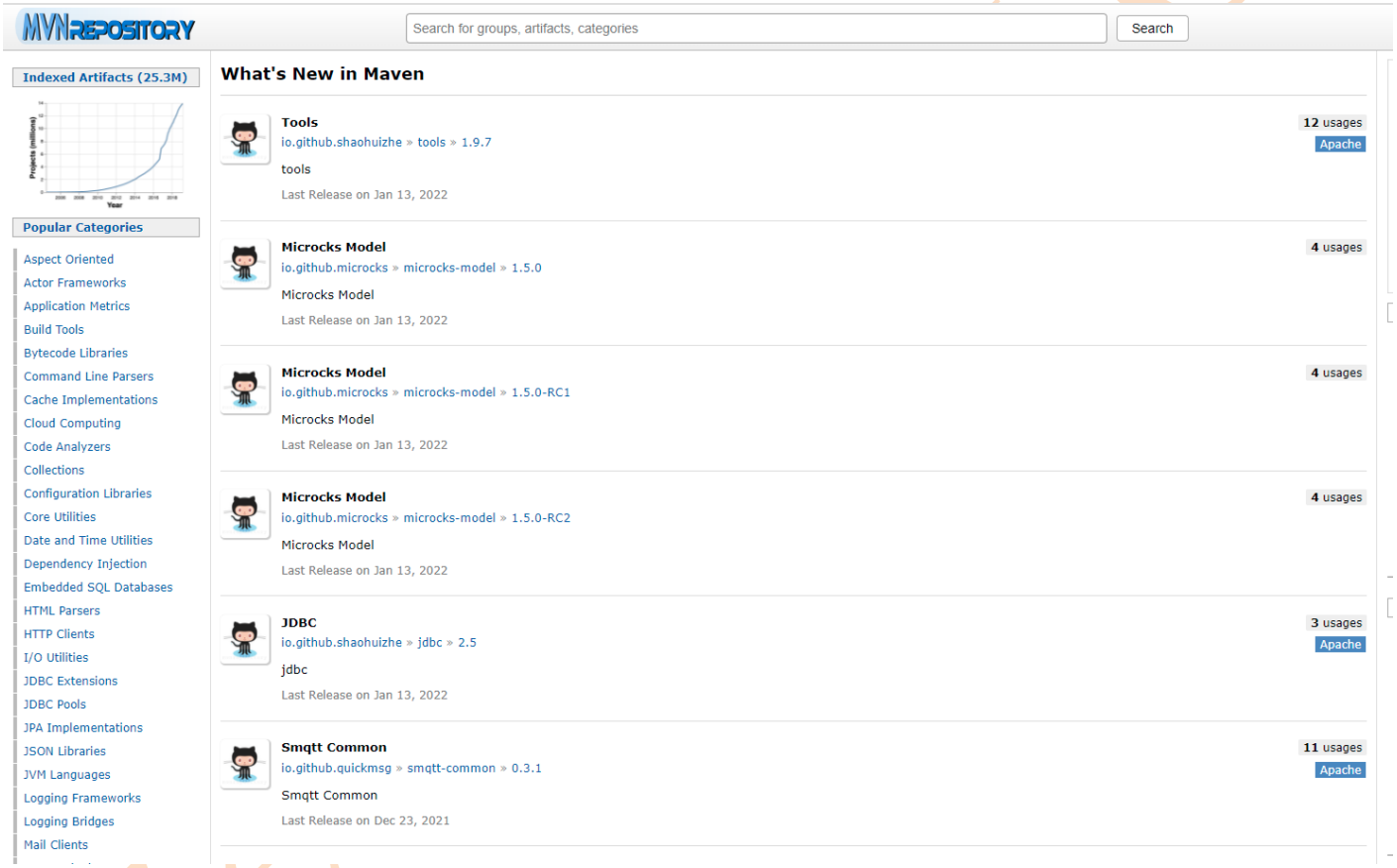
Репозитории.

При работе с Maven под репозиторием мы понимаем директорию, где хранятся все JAR, библиотеки, плагины и любые артефакты, которыми Maven может воспользоваться.

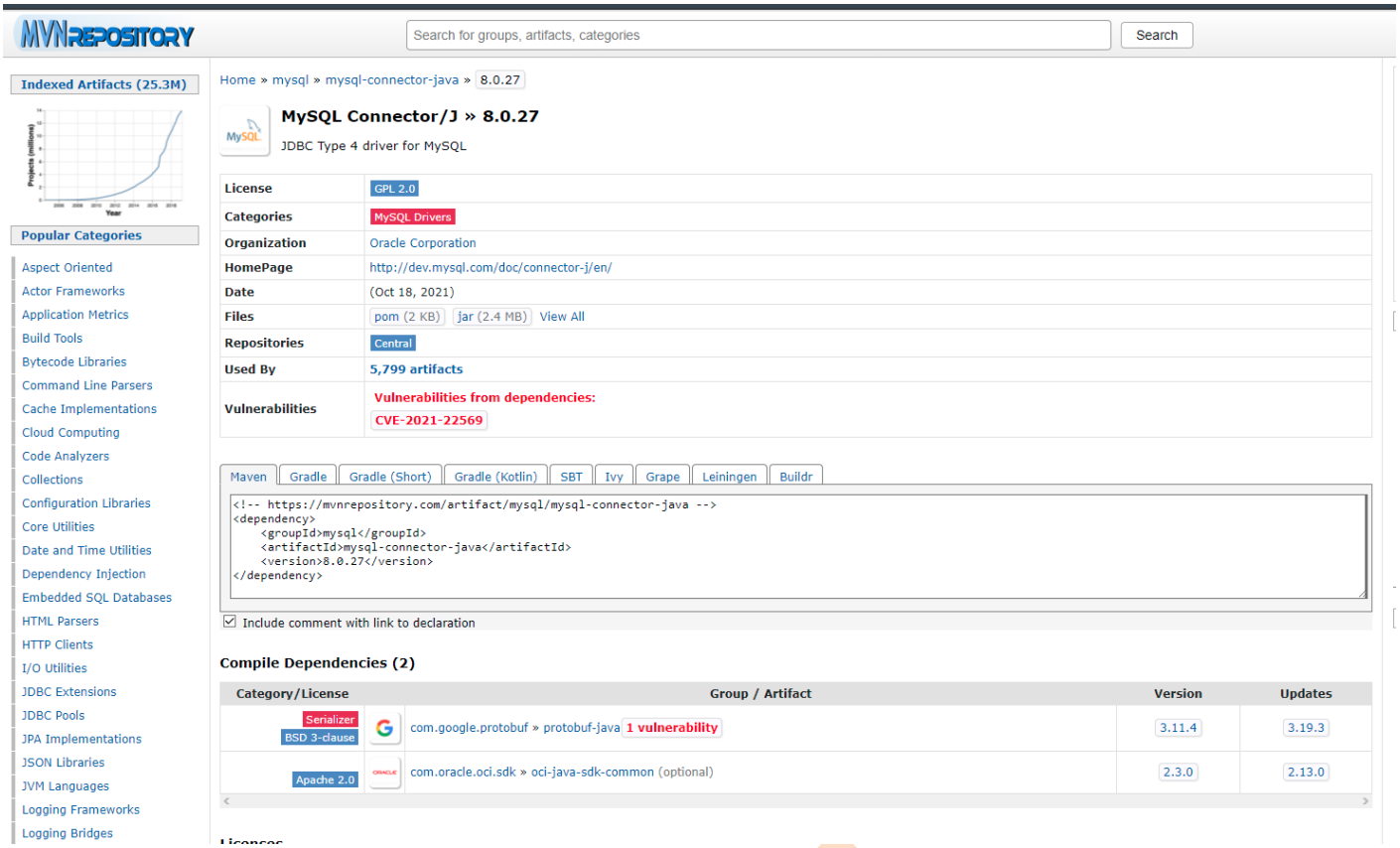
Существует три типа репозиториев Maven:

- **локальные (local)**
- **центральные (central)**
- **удалённые (remote)**

https://mvnrepository.com



Выбираем необходимую зависимость.
Далее:



Можем скопировать и использовать в своем проекте.
Все библиотеки будут доступны нам в коде автоматически.

Последовательность поиска зависимостей Maven

Когда мы выполняем команды сборки Maven, Maven начинает поиск библиотек зависимостей в следующей последовательности:

- **Шаг 1** — Поиск зависимости в локальном репозитории, если не найден, перейти к шагу 2, иначе выполнить дальнейшую обработку.

- **Шаг 2** — Поиск зависимости в центральном репозитории, если не найдено и упоминается / упоминается удаленный репозиторий / репозитории, затем перейдите к шагу 4. В противном случае он загружается в локальный репозиторий для дальнейшего использования.

- **Шаг 3** — Если удаленный репозиторий не был упомянут, Maven просто останавливает обработку и выдает ошибку (Невозможно найти зависимость).

- **Шаг 4** — Поиск зависимости в удаленном репозитории или репозиториях, если он найден, он загружается в локальный репозиторий для дальнейшего использования. В противном случае Maven останавливает обработку и выдает ошибку (Невозможно найти зависимость).

Плагины.

Если говорить в целом, то Maven – это фреймворк, который выполняет плагины. В этом фреймворке каждая задача, по сути своей, выполняется с помощью плагинов.

Плагины Maven используются для:

- **создания jar – файла**
- **создания war – файла**
- **компиляции кода файлов**
- **юнит-тестирования кода**
- **создание отчётов проекта**
- **создание документации проекта**

Типы плагинов

Существует два типа плагинов в Maven:

- **Плагины сборки**
  Выполняются в процессе сборки и должны быть конфигурированны внутри блока <build></build> файла **pom.xml**
- **Плагины отчётов**
  Выполняются в процесса генерирования сайта и должны быть конфигурированны внутри блока <reporting></reporting> файла **pom.xml**.

Вот список, наиболее используемых плагинов:

- **clean**
  Очищает цель после сборки. Удаляет директорию target.
- **compiler**
  Компилирует исходные Java файлы.
- **surefire**
  Запускает тесты JUnit. Создаёт отчёты о тестировании.
- **jar**
  Собирает JAR файл текущего проекта.
- **war**
  Собирает WAR файл текущего проекта.
- **javadoc**
  Генерирует Javadoc проекта.
- **antrun**
  Запускает набор задач **ant** из любой указанной фазы.

Для понимания того, как это работает на практике, рассмотрим следующий пример.

```
<profiles>
      <profile>
          <id>test</id>
          <activation>
              <file>
```

```
                    <missing>target/generated-sources/some/dir/com/iba/maven</missing>
                </file>
            </activation>
        </profile>
    </profiles>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-antrun-plugin</artifactId>
                <version>1.1</version>
                <executions>
                    <execution>
                        <id>id.clean</id>
                        <phase>compile</phase>
                        <goals>
                            <goal>run</goal>
                        </goals>
                        <configuration>
                            <tasks>
                                <echo>compile phase</echo>
                            </tasks>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

Выполняем команду:

```
C:\Users\User\Desktop\test_empty1>mvn compile
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------< test_empty:test_empty >-------------------------
[INFO] Building test_empty 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ test_empty ---
[WARNING] Using platform encoding (Cp1251 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] Copying 3 resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ test_empty ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-antrun-plugin:1.1:run (id.clean) @ test_empty ---
[INFO] Executing tasks
     [echo] compile phase
[INFO] Executed tasks
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  0.693 s
[INFO] Finished at: 2022-01-14T12:53:10+03:00
[INFO] ------------------------------------------------------------------------
```

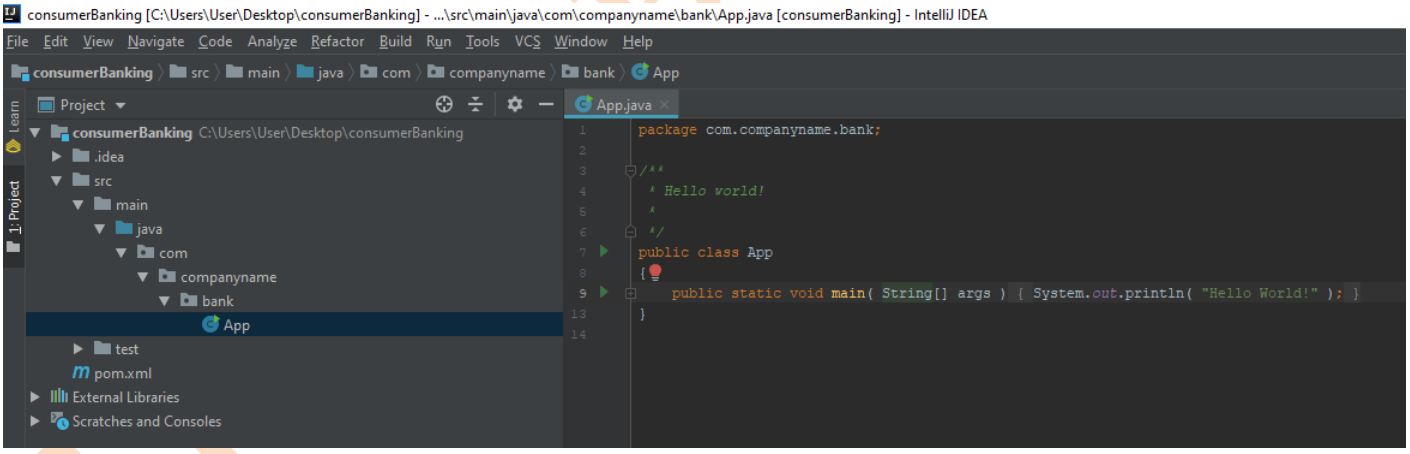
Добавьте Фазу(executions) compile.
Вызовите через консоль разные фазы и сравните результат в теге echo.

Создание проекта

Создадим проект на основе основных зависимостей. Откроем консоль и используем следующую команду:

```
C:\Users\User\Desktop>mvn archetype:generate -DgroupId=com.companyname.bank -DartifactId=consumerBanking -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.pom (703 B at 7.0 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-bundles/2/maven-archetype-bundles-2.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-bundles/2/maven-archetype-bundles-2.pom (1.5 kB at 16 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetype/maven-archetype-parent/1/maven-archetype-parent-1.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetype/maven-archetype-parent/1/maven-archetype-parent-1.pom (1.3 kB at 16 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/4/maven-parent-4.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/maven-parent/4/maven-parent-4.pom (10.0 kB at 111 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-quickstart/1.0/maven-archetype-quickstart-1.0.jar (4.3 kB at 57 kB/s)
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: C:\Users\User\Desktop
[INFO] Parameter: package, Value: com.companyname.bank
[INFO] Parameter: groupId, Value: com.companyname.bank
[INFO] Parameter: artifactId, Value: consumerBanking
[INFO] Parameter: packageName, Value: com.companyname.bank
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: C:\Users\User\Desktop\consumerBanking
[INFO] ----------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ----------------------------------------------------------------------------
[INFO] Total time:  3.422 s
[INFO] Finished at: 2022-01-14T13:24:37+03:00
```

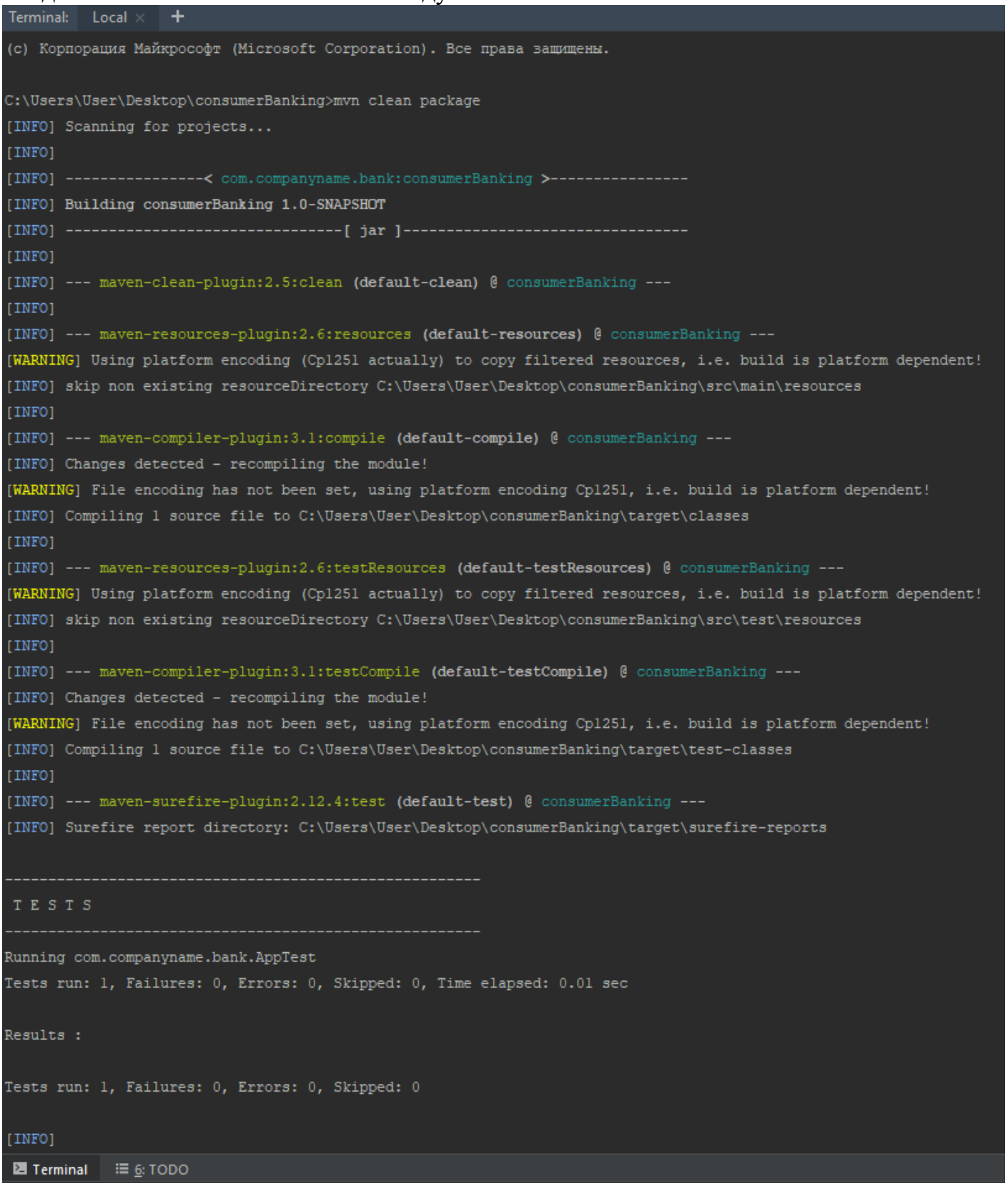
После чего перейдем в наш проект:



<u>Обратите внимание, что помимо класса, наш сборщик, создал файл тестирования проекта. Намекая нам о там, что все должно быть покрыто модульными тестами, о которых мы узнаем в середине нашего обучения.</u>

Сборка проекта:

Зайдем в консоль и выполним команду:



Обратите внимание, что тесты выполнены только потому, что в POM файле есть зависимость для тестирования!

Далее перейдите в папку classes и выполните команду:

```
C:\Users\User\Desktop\consumerBanking\target\classes>java com.companyname.bank.App
Hello World!

C:\Users\User\Desktop\consumerBanking\target\classes>
```

Наше приложение запустилось.

Откройте билд проекта и посмотрите, что специфического вы сможете обнаружить внутри.

Что можете сказать про последний собранный архив?

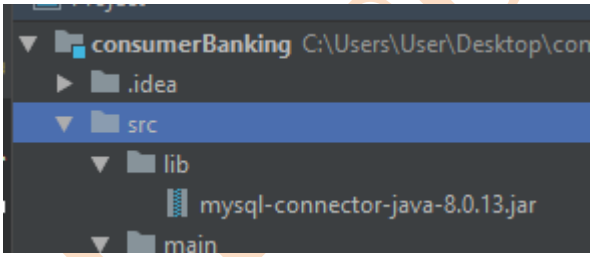
Внешние зависимости.

Ранее мы уже изучали управление зависимостями с помощью репозиториев. Но что, если необходимые файлы не найдены ни в центральном, ни на удалённом репозитории? Для решения этой проблемы используются внешние зависимости.

Рассмотрим такой пример.

Добавим в нам проект в папку **src** директорию **lib**.

Добавьте в эту директорию любой jar файл.



После чего давайте изменим наш пом файл

```
    <profiles>
        <profile>
            <id>test</id>
            <activation>
                <file>
                    <missing>target/generated-sources/some/dir/com/iba/maven</missing>
                </file>
            </activation>
        </profile>
    </profiles>
    <dependencies>
```

```
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>3.8.1</version>
            <scope>test</scope>
        </dependency>

        <!-- External dependency -->
        <dependency>
            <groupId>mysql-connector-java-8.0.13.jar</groupId>
            <artifactId>mysql-connector-java-8.0.13.jar</artifactId>
            <scope>system</scope>
            <version>1.0</version>
            <systemPath>C:/Users/User/Desktop/consumerBanking/src/lib/mysql-connector-
java-8.0.13.jar</systemPath>
        </dependency>

    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-antrun-plugin</artifactId>
                <version>1.1</version>
                <executions>
                    <execution>
                        <id>id.clean</id>
                        <phase>compile</phase>
                        <goals>
                            <goal>run</goal>
                        </goals>
                        <configuration>
                            <tasks>
                                <echo>compile phase</echo>
                            </tasks>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

После наших правок, проект сможет обработать нашу библиотеку.

Шаблоны проектов.

С помощью концепции **Архитипов** Maven обеспечивает большое количество различных шаблонов проектов (более 600).

Архитип

Архитип – это плагин Maven, задача которого состоит в создании структуры проекта по определённому шаблону. Рассмотри архитип quickstart и создадим с его помощью java-приложение.

Попробуем еще вариант создания архетипа:
Перейдем в консоль и выполним команду mvn archetype:generate

Далее нажимаем цифру 7.
После чего выбираем вид архетипа: 1.
Далее нужно дать метаданные нашему проекту.

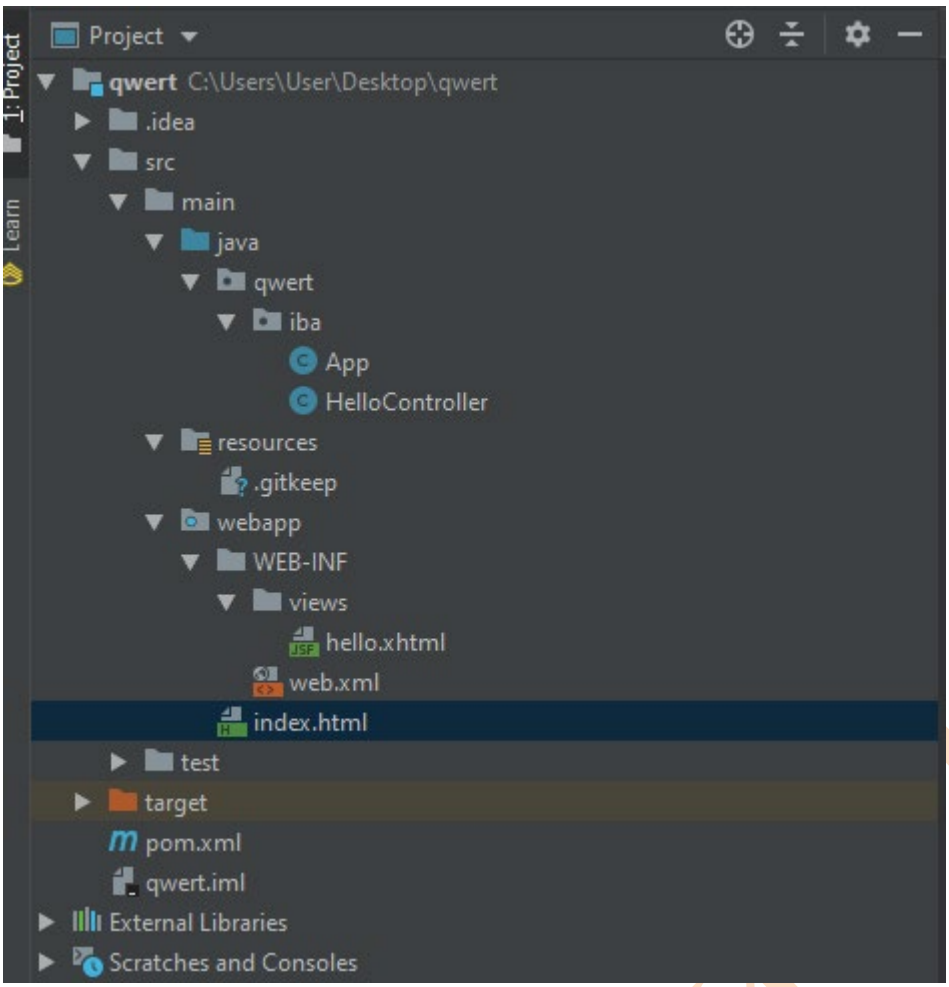
```
Choose a number or apply filter (format: [groupId:]artifactId, case sensitive contains): 1854: 7
Choose am.ik.archetype:mvc-1.0-blank-archetype version:
1: 1.0.0-m01
2: 1.0.0-m02
Choose a number: 2: 1
Downloading from central: https://repo.maven.apache.org/maven2/am/ik/archetype/mvc-1.0-blank-archetype/1.0.0-m01/mvc-1.0-blank-archetype-1.0.0-m01.pom
Downloaded from central: https://repo.maven.apache.org/maven2/am/ik/archetype/mvc-1.0-blank-archetype/1.0.0-m01/mvc-1.0-blank-archetype-1.0.0-m01.pom (2.6 kB at 20 kB/s)
Downloading from central: https://repo.maven.apache.org/maven2/am/ik/archetype/mvc-1.0-blank-archetype/1.0.0-m01/mvc-1.0-blank-archetype-1.0.0-m01.jar
Downloaded from central: https://repo.maven.apache.org/maven2/am/ik/archetype/mvc-1.0-blank-archetype/1.0.0-m01/mvc-1.0-blank-archetype-1.0.0-m01.jar (4.0 kB at 60 kB/s)
Define value for property 'groupId': eximple
Define value for property 'artifactId': eximple_1
Define value for property 'version' 1.0-SNAPSHOT: : 1.0-SNAPSHOT
Define value for property 'package' eximple: : eximple.new.iba
Confirm properties configuration:
groupId: eximple
artifactId: eximple_1
version: 1.0-SNAPSHOT
package: eximple.new.iba
 Y: : y
```

В конце подтверждаем Y

В результате

```
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Archetype: mvc-1.0-blank-archetype:1.0.0-m02
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: groupId, Value: eximple
[INFO] Parameter: artifactId, Value: eximpleNew
[INFO] Parameter: version, Value: 1.0 SNAPSHOT
[INFO] Parameter: package, Value: eximple.new.iba
[INFO] Parameter: packageInPathFormat, Value: eximple/new/iba
[INFO] Parameter: package, Value: eximple.new.iba
[INFO] Parameter: version, Value: 1.0 SNAPSHOT
[INFO] Parameter: groupId, Value: eximple
[INFO] Parameter: artifactId, Value: eximpleNew
[INFO] Project created from Archetype in dir: C:\Users\User\Desktop\eximpleNew
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  48.630 s
[INFO] Finished at: 2022-01-14T14:15:18+03:00
[INFO] ------------------------------------------------------------------------
```

С глобального репозитория мы получили часть проекта:


Project
1: Project
Learn
qwert C:\Users\User\Desktop\qwert
.idea
src
main
java
qwert
iba
App
HelloController
resources
.gitkeep
webapp
WEB-INF
views
hello.xhtml
web.xml
index.html
test
target
pom.xml
qwert.iml
External Libraries
Scratches and Consoles